How it works

Introduction

At a very simple level, the Apache HTTP Server (called 'httpd') works like any other server - it creates a 'listening' socket on which it can await connections from clients. When a connection arrives, it reads the request submitted by the client, works out what to do with it (usually retrieve a file), and returns the response back to the client. Alas, the reality is somewhat more complicated, since httpd must handle many such requests from multiple clients concurrently in a highly efficient manner. It also has to handle different types of requests - plain file requests, cgi requests, proxy requests, authentication, and so on. This page describes some of the more pertinent aspects of the Apache httpd architecture and internal workings.

Architecture

The Apache HTTP Server architecture is based on a relatively simple concept called the Multi-Process Module, or MPM. This is used to control how the Apache HTTP Server (called 'httpd') handles requests from clients. In most cases, a standard unix installation of the Apache HTTP server defaults to what is called a ***prefork*** model. Most of the contents of this Wiki will assume this is the case unless specifically documented otherwise. The main alternative to the pre-fork MPM is the ***worker*** MPM. Both these are described in more detail below.

Prefork MPM

In the prefork model, a single manager, or parent process is used to control a pool of child processes. The child processes are responsible for handling incoming requests from the clients, while the parent process is responsible for monitoring the health of the child processes, creating new child processes when needed, or terminating child processes when no longer needed. Each child process is single threaded, and can handle just one request concurrently. The maximum number of requests than can be processed concurrently is simply equal to the number of child processes.

Advantages

It is thread safe (by virtual of not using them!).

Misbehaving request does not impact other requests.

Disadvantages

Generally uses more resource for a given load than the worker MPM.

Worker MPM

In the worker model, there is again a single manager, or parent process that is used to control a pool of child processes, and the child processes are responsible for handling incoming requests. However, unlike the prefork model, in the worker model, each child process is multi threaded and can handle multiple requests concurrently. The maximum number of requests that can be processed concurrently is the number of child processes multiplied by the number of threads per child process.

Advantages

Generally uses less resources for a given load than the prefork MPM.

Disadvantages

Any modules will need to be thread safe. For example, mod_php may not be thread-safe, depending upon the third-party libraries it is linked with.

Request Processing

Regardless of the MPM type of an Apache server instance, once a request is accepted for processing, it goes through a processing pipeline consisting of a number of distinct phases. Each phase may itself consist of a number of different steps. In order, the phases are:

Request Parsing

In this phase, the request URL is unescaped and converted to an absolute path, then translated to a file system path.

Security Checking

In this phase, any security directives are applied to the identified resource. Only if the user has the necessary authorization does the request processing continue.

Preparation Phase

In this phase, minor changes to the request are made - quite often just a case of filling in additional information needed by the subsequent handling phase. This might include content and mime types.

Handler Phase

This is the phase where the content generation generally takes place.

Most of the steps within each phase have default implementions which can be overridden or extended by custom modules.

Modules

In addition to the MPM, Apache httpd supports ***modules***. A module is a loadable 'chunk' of code that is usually designed to implement a specific set of functionality - for example, proxying or compression. Modules are loaded and configured when httpd starts up, and install functions into various parts of the process to provide the required functionality. With release 2 of Apache httpd, most of the functionality of the server is implemented using modules - everything from the basic request handling, aliases, URL re-writing, CGI execution, SSL, authentication, etc.